2

WORKING MATERIAL

for the lectures of

Stanley S. Wainer

Computability - Logical and Recursive Complexity

International Summer School

on

LOGIC, ALGEBRA AND COMPUTATION

Marktoberdorf, Germany, July 25 - August 6, 1989

This Summer School is organized under the auspices of the Technische Universität München and is sponsored by the NATO Science Commitee as part of the 1988 Advanced Study Institutes programme, partial support for the conference was provided by the European Research Office and the National Science Foundation and by various industrial companies.

Computability - Logical and Recursive Complexity

Stan Wainer (Leeds UK).

Introduction.

The basis of this short course of 4 lectures is the strong analogy between programs and proofs (of their specifications). The main theme is the classification of computable number-theoretic functions according to the logical complexity of their formal specification or termination proofs. A significant sub-branch of mathematical logic has grown around this theme since the 1950's and new ideas are presently giving rise to further developments. The methods employed are chiefly those from proof theory, particularly "normalization" as presented in the lectures of H. Schwichtenberg, and "ordinal assignments". Since program-termination corresponds to well-foundedness of computation trees, it is hardly surprising that transfinite ordinals and their constructive representations play a crucial role, measuring the logical complexity of programs and of the functions which they compute. (KR) ←

Note that we are concerned here solely with computable (i.e. recursive) number-theoretic functions

$$f : N^k \longrightarrow N$$

where $N = \{0, 1, 2, 3, \ldots\}$. Of course computation in general involves many different sorts of data-structures, but among these N is of basic foundational importance, its theory is rich and well-developed, and it therefore best illustrates the underlying ideas linking computation theory with proof theory.

When asked for the main outstanding problem in the theory of recursion, Gödel is reported to have answered: classify the recursive functions!

We are still trying; but what follows is a report on some partial success toward the classification problem.

The only way we know of constructing "new" recursive functions is by "diagonalizing" through the class already at hand. Thus in order to define a recursive function which is not provably specifiable in a given formal theory, we need to diagonalize through those specifications which are provable. We also need to know they are true and this presupposes the consistency of the theory. The lesson is that we can successfully (usefully) classify the recursive functions which are provably specifiable in formal theories where "ordinal analyses of consistency strength" are established. This represents a very large class of recursive functions! - but by no means all!

§1. Programs and Proofs.

A number-theoretic *program-specification* is a so-called Π^0_2- or $\forall\exists$- statement of the form

$$\forall \text{input } x \; \exists \text{output } y \; R(x, y)$$

where $R(x, y)$ expresses the fact that the output y is "correctly related" to the input x.

A number-theoretic program $p(x) \Leftarrow (\ldots\ldots)$ *satisfies* the above specification if for every input $x \in N$, the program computes an output $y = p(x)$ such that $R(x, p(x))$ is true.

Note that we are specifying *totally-defined* functions, i.e. specification $\rightarrow$ termination.

Question 1. How does a given formal theory of arithmetic restrict the class of programs it can provably specify?

Question 2. How can we synthesize programs from their specification-proofs in a given formal theory?

Illustration (Kreisel, Parsons, Mints, Takeuti...). From specification-proofs in the theory of Σ^0_1-Induction, we can synthesize *primitive recursive* programs.

Rather than giving a complete proof of this old established result, we shall merely indicate the main proof-theoretic ideas underlying it in order to show up the clear and natural correspondence which exists, between proof-rules on one hand and recursive programming constructs on the other.

First we must describe the theory of Σ_1^0-Induction. It is a certain sub-theory of full first-order Peano Arithmetic PA, which can be formalized in the following way (convenient for proof-theoretical analysis):

We derive not single formulas A, B, C standing alone, but in general finite sets of formulas $\Gamma = \{A_0, A_1, \ldots, A_m\}$ or $\Delta = \{B_0, B_1, \ldots, B_k\}$, the intuitive meaning being (A_0 or A_1 or ... or A_m) etcetera. The single formula B is identified with the singleton set $\{B\}$ but we always omit $\{ , \}$ in such a case and simply write B. Thus Γ, A stands for $\Gamma \cup \{A\}$ and Δ, B, C stands for $\Delta \cup \{B\} \cup \{C\}$ etc.

The only terms of PA are the constant 0, the variables $x, y, z, \ldots$ and those which can be constructed from these by repeated applications of given function symbols such as S (for successor). There may be others.

The atomic or prime formulas are constructed from terms by applying relation symbols which come in dual pairs: R and its complement $\bar{R}$.

In this way negation becomes a defined symbol using De Morgan's laws thus: $\neg R \equiv \bar{R}$, $\neg \bar{R} \equiv R$, $\neg(A \wedge B) \equiv \neg A \vee \neg B$, $\neg(A \vee B) \equiv \neg A \wedge \neg B$, $\neg \exists x \equiv \forall x \neg$, $\neg \forall x \equiv \exists x \neg$. Implies is then also defined as $(A \to B) \equiv \{\neg A, B\}$.

Full Peano Arithmetic has

Logical axioms $\Gamma, \neg A, A$ where A is atomic.

Arithmetical axioms Γ, Δ where Δ is a set of prime formulas defining one of the given elementary relations. For example, to define the relation $R(x,y,z) \equiv$ "$x+y=z$" we need axioms:

$$\Gamma, R(x,0,x)$$
$$\Gamma, \bar{R}(x,y,z), R(x, Sy, Sz)$$
$$\Gamma, \bar{R}(x,y,z), \bar{R}(x,y,z'), z=z'$$

and all substitution instances of them.

Logical rules

$(\wedge)$	$\vdash \Gamma, (A \wedge B)$	if $\vdash \Gamma, A$ and $\vdash \Gamma, B$
$(\vee)$	$\vdash \Gamma, (A \vee B)$	if $\vdash \Gamma, A$ or $\vdash \Gamma, B$
$(\forall)$	$\vdash \Gamma, \forall x A(x)$	if $\vdash \Gamma, A(y)$ – y not free in Γ.
$(\exists)$	$\vdash \Gamma, \exists x A(x)$	if $\vdash \Gamma, A(t)$ – some term t.
(Cut)	$\vdash \Gamma$	if $\vdash \Gamma, C$ and $\vdash \Gamma, \neg C$

The Induction rule

(Ind) $\vdash \Gamma, A(x)$ if $\vdash \Gamma, A(0)$ and $\vdash \Gamma, \neg A(x), A(Sx)$

where x is not a free variable in Γ

The Theory of Σ^0_1-Induction has the same axioms and logical rules as PA above, but the rule of induction is restricted to apply not to all formulas $A(x)$ but only those in Σ^0_1- or $\exists$-form; for example $A(x) \equiv \exists y R(x,y)$

Now we have set up the formal system, let us return to the Illustration:

Suppose we have a proof in Σ^0_1-Induction, of a specification:

$$\forall x \exists y R(x,y)$$

This is not a Σ^0_1-formula because of the $\forall x$ at the front. However it is a simple matter to check through all the rules and axioms and see that from any proof of $\Gamma, \forall x A(x)$ we can extract a proof of $\Gamma, A(z)$ as long as z is not a free variable in $\Gamma, \forall x A(x)$

Therefore from the original specification proof we can extract a proof of

$$\exists y R(x,y)$$

with x now a free variable.

Now what was the final rule applied in proving

$$\exists y\, R(x,y) \qquad ?$$

It could only have been an $\exists$-rule, or an Induction-rule, or a Cut-rule. We therefore have just 3 cases to consider, though in fact the Cut-case splits into 2 subcases.

In each case we exhibit a typical (goal-directed) proof of

$$\exists y\, R(x,y)$$

and from the proof we can immediately see how to extract / synthesize a program

$$p(x) \Leftarrow (\ldots\ldots)$$

which satisfies the specification.

The $\exists$-case

$$\begin{tikzcd} \exists y\, R(x,y) \\ R(x,t(x)) \arrow[u, "\exists"'] \end{tikzcd} \qquad \text{some term } t.$$

Here, the witnessing term already satisfies R for input x, so the extracted program is.

$$\boxed{\text{Assignment.} \quad p(x) \longleftarrow t(x)}$$

The Induction - case

$$\begin{tikzcd}
& \exists y R(x,y) & \\
\exists y R(0,y) \arrow[ur] & & \forall y \neg R(x,y), \exists y R(x+1,y) \arrow[ul] \\
R(0,t_0) \arrow[u, "\exists"'] & & \neg R(x,y), \exists y R(x+1,y) \arrow[u, "\forall"] \\
& & \neg R(x,y), R(x+1,t_1(x,y)) \arrow[u, "\exists"']
\end{tikzcd}$$

From this we read off witnessing terms t_0 for the base case, and $t_1(x,y)$ for the successor case - provided that y is already assumed to satisfy R at x! The synthesized program is therefore

Primitive Recursion
$$\begin{aligned} p(0) &\Longleftarrow t_0 \\ p(x+1) &\Longleftarrow t_1(x, p(x)) \end{aligned}$$

The Cut - case

$$\begin{tikzcd}
& \exists y R(x,y) & \\
C , \exists y R(x,y) \arrow[ur] & & \neg C , \exists y R(x,y) \arrow[ul]
\end{tikzcd}$$

The problem here is that the cut-formula C may be arbitrarily complex and we are stuck unless we can somehow restrict its complexity.

Proof theory at this point comes to our rescue!

Cut Elimination Theorem (Gentzen, ...)
Though we cannot eliminate cuts completely from proofs in Σ_1^0-Induction, we can always effectively transform any such proof into another wherein all cut-formulas are Σ_1^0.

So assume this done at the outset. Then we know that the cut-formula C has the form

$$C \equiv \exists z\, Q(x,z)$$

There are now essentially two possible ways in which the left-hand branch of our Cut-proof of $\exists y\, R(x,y)$ could come about:

Sub-case 1

$$\begin{tikzcd}
 & \exists y\, R(x,y) & \\
\exists z\, Q(x,z),\ \exists y\, R(x,y) \arrow[ur] & & \forall z\, \neg Q(x,z),\ \exists y\, R(x,y) \arrow[ul] \\
\exists z\, Q(x,z)\ ,\ R(x,t_0(x)) \arrow[u, "\exists"'] & & \neg Q(x,z)\ ,\ \exists y\, R(x,y) \arrow[u, "\forall"] \\
Q(x,t_1(x)),\ R(x,t_0(x)) \arrow[u, "\exists"] & & \neg Q(x,z)\ ,\ R(x,t_2(x,z)) \arrow[u, "\exists"']
\end{tikzcd}$$

Thus either the term $t_1(x)$ witnesses Q at x, in which case (putting $z = t_1(x)$) the term $t_2(x,z)$ witnesses R, *or else* $t_1(x)$ fails to witness Q in which case $t_0(x)$ witnesses R. Therefore we can extract the following program satisfying R:

Conditional. $p(x) \Leftarrow$ *if* $Q(x, t_1(x))$ *then* $t_2(x, t_1(x))$ *else* $t_0(x)$

Sub-case 2

It is possible that $\exists y R(x,y)$ might be a "side-formula" in the left-hand branch and that that branch actually contains a *proof* of

$$C \equiv \exists z\, Q(x,z).$$

In this case we can assume inductively that a program $q(x) \Leftarrow (\ldots)$ satisfying the sub-specification C has already been synthesized!

$$\begin{tikzcd}
& \exists y\, R(x,y) & \\
\exists z\, Q(x,z) \arrow[ur] \arrow[d, Rightarrow] & & \forall z \neg Q(x,z),\ \exists y\, R(x,y) \arrow[ul] \\
Q(x, q(x)) & & \neg Q(x,z),\ R(x, t(x,z)) \arrow[u, Rightarrow, "\forall", "\exists"']
\end{tikzcd}$$

From this situation we extract the program

Composition. $p(x) \Leftarrow t(x, q(x))$

$\square$

Corollary

The non-primitive recursive Ackermann function is not provably specifiable in Σ^0_1-Induction.

Actually it requires Π^0_2-Induction, the "next" sub-theory of PA in which the Induction rule is restricted to Π^0_2-formulae.

How can we make some computational use of this next higher level of induction ?

Illustration

Suppose f is some given function and let

$$\mathrm{It}(x, y, z) \quad \text{mean} \quad z = f^x(y) = f \circ \ldots \circ f(y).$$

We can derive various programs satisfying
$$\forall x, y \, \exists z \, \mathrm{It}(x, y, z)$$
from the following Π^0_2-inductive proof:

$$\begin{tikzcd}
 & \forall x \, \forall y \, \exists z \, \mathrm{It}(x,y,z) & \\
 & \forall y \, \exists z \, \mathrm{It}(x,y,z) \arrow[u, "\forall"'] & \\
\forall y \, \exists z \, \mathrm{It}(0,y,z) \arrow[ur] & & \exists y \, \forall z \, \neg \mathrm{It}(x,y,z), \forall y \, \exists z \, \mathrm{It}(x+1,y,z) \arrow[ul] \\
\exists z \, \mathrm{It}(0,y,z) \arrow[u, "\forall"'] & & \forall z \, \neg \mathrm{It}(x,t_1,z), \exists z \, \mathrm{It}(x+1,y,z) \arrow[u, "\exists", "\forall"'] \\
\mathrm{It}(0,y,t_0) \arrow[u, "\exists"'] & & \neg \mathrm{It}(x,t_1,z), \mathrm{It}(x+1,y,t_2) \arrow[u, "\forall", "\exists"']
\end{tikzcd}$$

To complete this proof the theorem-prover has to produce terms t_0, t_1, t_2 so that the following hold:

$$It(0, y, t_0)$$
$$It(x, t_1, z) \longrightarrow It(x+1, y, t_2).$$

Obviously t_0 has to be the term "y" since $f^0(y) = y$. But there are two possible ways of choosing (t_1, t_2), giving rise to two different synthesized programs.

I. Choose $t_1 \equiv y$ and hence $t_2 \equiv f(z)$.

The extracted program is then the standard recursive definition of iteration:

$$\boxed{\begin{array}{lcl} p(0, y) & \Leftarrow & y \\ p(x+1, y) & \Leftarrow & f(p(x, y)) \end{array}}$$

As this is a simple primitive recursion it could already have been proved in Σ_1^0-Induct.

II. Choose $t_2 \equiv z$ and hence $t_1 \equiv f(y)$.

The extracted program is now tail-recursive:

$$\boxed{\begin{array}{lcl} p(0, y) & \Leftarrow & y \\ p(x+1, y) & \Leftarrow & p(x, f(y)) \end{array}}$$

so it could be reformulated as a while-loop:

$$\boxed{\underline{\text{while}}\ x \neq 0\ \underline{\text{do}}\ x := x-1\ ;\ y := f(y)\ \underline{\text{od}}}$$

The reason for program II being tail-recursive is that we insisted on choosing $t_2 \equiv z$. This meant that we had to change t_1 from y to $f(y)$. Thus program II involves substituting $f(y)$ for the original parameter y. A moments thought will convince the reader that in order to prove that program II is defined one needs to show, by induction, that for each x the following holds

$$\forall y \exists z \, (p(x,y) = z)$$

and the $\forall y$ cannot be removed here! Thus Π^0_2-Induction was essential in deriving the tail-recursive / while - program, because without it the term t_1 would not have arisen and we would then not have had the freedom to replace y by $f(y)$.

Thus Π^0_2-Induction is a useful framework for program transformations from primitive recursions into while-loops. We have the ratios

$$\frac{\mathrm{I}}{\mathrm{II}} = \frac{\text{Prim. Rec.}}{\text{While-loop}} = \frac{\Sigma^0_1\text{-Ind.}}{\Pi^0_2\text{-Ind.}}$$

The cost of the transformation is an increase in induction-complexity of the specification proof, from Σ^0_1 to Π^0_2!

Can we use proof theory to further analyse, and quantify the complexity of this transformation?

§2. Terminating Recursions & Well Foundedness.

"Recursion on the variable a" is the definition of a function value $f(a,x)$ in terms of other values $f(p(a,x), s(x))$ where $p(a,x)$ is in some sense a predecessor of a. For the recursion to terminate the sequences

$a\ ,\ p(a,x)\ ,\ p(p(a,x),s(x))\ ,\ p(p(p(a,x),s(x)),s^2(x))\ldots$
must "hit bottom" after finitely many steps. i.e. predecessors should form a well-founded tree. Furthermore we may as well assume that if a and a' have identical trees of predecessors, then they are themselves identical, i.e. $a = a'$.

For the predecessors to be linearly (hence well-) ordered, each $p(a,x)$ should be a predecessor of $p(a,y)$ whenever $x < y$. The ordinal height of the well ordering should then provide a useful measure of complexity of the recursion.

These informal ideas motivate the definitions below. Note that we may consider $p(a,x)$ as a sequence of predecessors $P_0(a), P_1(a), P_2(a), \ldots$ one for each value of x where $P_x(a) = p(a,x)$.

2.1 Definition. A well-founded recursion structure (WFRS) is a structure $A = \langle A, 0_A, P_0, P_1, \ldots \rangle$ where for each $x \in N$, $P_x : A \to A$ with $P_x(0_A) = 0_A$ and

$(*)\ \forall a \in A.\forall x \in N.\ \forall s : N \to N.\ \exists n\ (P_{s^n(x)} \cdots P_{s^2(x)} P_{s(x)} P_x(a) = 0_A)$

$(**)\ \forall a, b \in A\ (\ \forall x\ (P_x(a) = P_x(b)) \longrightarrow a = b\ )$.

2.2 Definition A WFRS *with successor* is one for which there is a function $a \mapsto a +_A 1$ such that for each $x \in N$ and each $a \in A$, $P_x(a +_A 1) = a$.

2.3 Definition A *well-ordered recursion structure*, or WORS for short, is a WFRS with successor in which, for every $a \in A$ and all $x, y \in N$,

$$x < y \longrightarrow \exists m \, (P_x(a) +_A 1 = P_y^m(a))$$

2.4 Definition. *Recursion* over a WFRS is any scheme

$$\begin{cases} f(0_A; x_1 \ldots x_k) = g_0(x_1 \ldots x_k) \\ f(a; x_1 \ldots x_k) = \text{Term}(g_1 \ldots g_m, f_{P(a)}; x_1 \ldots x_k) \end{cases}$$

where $g_0, g_1 \ldots g_m$ are given functions and for each non-zero $a \in A$, and all $z, y_1 \ldots y_k \in N$,

$$f_{P(a)}(z, y_1 \ldots y_k) = f(P_z(a); y_1 \ldots y_k).$$

A number-theoretic function $h : N^k \to N$ is said to be defined (from given functions $g_0, g_1 \ldots g_m$) by *A-recursion* as above, if there is some fixed element $a_h \in A$ such that for all $x_1 \ldots x_k \in N$,

$$h(x_1 \ldots x_k) = f(a_h; x_1 \ldots x_k).$$

Remark. a_h measures the level of complexity of h, as defined by the given A-recursion. Note that by condition (*), *every A-recursion terminates*.

2.5 Definition REC(A) denotes the smallest class of number-theoretic functions containing all "elementary" ones and closed under explicit definitions and all A-recursions.

What we would like to have now, is a convenient uniform way of representing well-founded recursion structures (with successor), so that we can hopefully compare them, and develop ways of effectively computing one from another. It is at this point that we need to introduce a constructively-motivated version of transfinite ordinals.

2.6 Definition. The set Ω of countable tree-ordinals $\alpha, \beta, \gamma, \ldots$ is inductively generated by the rule:

$$\begin{array}{lll} \alpha \in \Omega & \text{if} & \alpha = 0 \\ & \text{or} & \alpha = \beta+1 = \beta \cup \{\beta\} \text{ for some } \beta \in \Omega \\ & \text{or} & \alpha \text{ is a function from } N \text{ into } \Omega. \end{array}$$

If α is a function from N into Ω we denote its value at $x \in N$ by α_x and write "$\alpha = \sup \alpha_x$"

2.7 Definition The predecessor functions $P_0, P_1, \ldots$ on Ω are defined as follows (following Cichon)

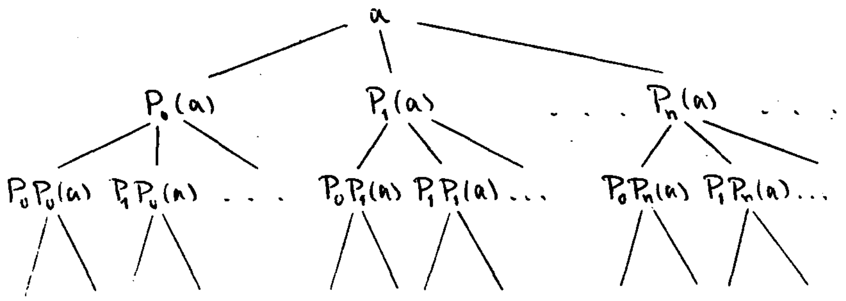
$$P_x(0) = 0 \quad , \quad P_x(\alpha+1) = \alpha \quad , \quad P_x(\sup \alpha_n) = P_x(\alpha_x).$$

Thus if $A \subset \Omega$ is non-empty, closed under predecessors and $+1$, then $\langle A, 0, P_0, P_1, \ldots \rangle$ is a WFRS with successor provided (**) holds.

2.8 Representation Theorem. Every WFRS with $+_a 1$ is isomorphic to a subset $\Omega(A) = \{o(a) : a \in A\}$ of Ω where the map $o : A \to \Omega$ is given by:

$$o(0_A) = 0 \quad , \quad o(a +_a 1) = o(a) + 1 \quad , \quad o(a) = \sup_x (o(P_x^A a) + 1) \text{ ow.}$$

<u>Proof</u>. Given any WFRS A, the well-foundedness condition (*) allows us to prove properties of $a \in A$ by induction up the tree of predecessors of a.


a
$P_0(a)$
$P_1(a)$
$\dots$ $P_n(a)$ $\dots$
$P_0P_0(a)$ $P_1P_0(a)$ $\dots$
$P_0P_1(a)$ $P_1P_1(a)$ $\dots$
$P_0P_n(a)$ $P_1P_n(a)$ $\dots$

For example we can easily show that the map $o : A \to \Omega$ is 1-1, i.e. $o(a) = o(b) \longrightarrow a = b$.

Case $a = 0_A$: then $o(a) = 0$ so $o(b) = 0$ and hence $b = 0_A$.

Case $a = a' +_A 1$: then $o(a) = o(a') + 1 = o(b)$, so $b = b' +_A 1$ and therefore $o(a') + 1 = o(b') + 1$, hence $o(a') = o(b')$. By the induction hypothesis applied to $a' = P_0^{\wedge}(a)$ we therefore have $a' = b'$ and consequently $a = b$ by (**).

Case otherwise: then $o(a) = \sup(o(P_x^{\wedge}(a)) + 1) = o(b)$ so b is neither 0_A nor a successor $b' +_A 1$, and hence $o(b) = \sup(oP_x^{\wedge}(b) + 1)$ by definition. Consequently for every x, $oP_x^{\wedge}(a) + 1 = oP_x^{\wedge}(b) + 1$ and $oP_x^{\wedge}(a) = oP_x^{\wedge}(b)$ so $P_x^{\wedge}(a) = P_x^{\wedge}(b)$ by the induction-hypothesis applied to $P_x^{\wedge}(a)$. Then $a = b$ follows immediately from (**).

It only remains to check that predecessors P_x are preserved by the map $\circ$, i.e. that

$$P_x(\circ(a)) = \circ(P_x^A(a))$$

This is immediate however. If $a = 0_A$ or $a = a' +_A 1$ then it follows directly from the definition of $\circ$. Otherwise we have $\circ(a) = \sup(\circ(P_x^A(a)) + 1)$, so

$$P_x(\circ(a)) = P_x(\circ(P_x^A(a)) + 1) = \circ(P_x^A(a)) \quad \square$$

Note The Representation Theorem tells us that Ω should provide all the recursion structures we are likely to need, so henceforth we will work inside Ω.

If we start with a well-ordered recursion structure A as given by Def. 2.3, and if $a \in A$ is neither 0_A nor a successor $a' +_A 1$, then $\circ(a) = \sup \alpha_x$ where $\alpha_x = \circ(P_x^A(a) +_A 1)$ and

$$x < y \longrightarrow \exists m\,(P_x(a) +_A 1 = P_y^m(a)).$$

Therefore

$$x < y \longrightarrow \exists m\,(\alpha_x = P_y^m(\sup \alpha_x)).$$

2.9 Definition. A tree-ordinal $\alpha \in \Omega$ is said to be structured if for every "sub-tree" of α of the form $\lambda = \sup \lambda_x$ we have for all $x, y \in N$,

$$x < y \longrightarrow \exists m\,(\lambda_x = P_y^m(\lambda))$$

The set of all structured tree-ordinals is denoted Ω^s.

2.10 Definition For each $\alpha \in \Omega$ and every $x \in N$, set

$$\alpha[x] = \{P_x(\alpha), P_xP_x(\alpha), \ldots, P_x^n(\alpha) = 0\}.$$

Thus, by the definition of the predecessor functions on Ω we have for each $x \in N$,

$$0[x] = \phi \ , \quad \alpha+1\,[x] = \alpha[x] \cup \{\alpha\} \ , \quad \sup\alpha_n[x] = \alpha_x[x].$$

<u>2.11 Definition</u>. Let $<$ denote the <u>sub-tree partial ordering</u> on Ω where $\beta \not< 0$, $\beta < \alpha+1$ iff $\beta \leq \alpha$, and $\beta < \sup\alpha_n$ iff $\beta \leq \alpha_k$ for some k.
For each fixed $\alpha \in \Omega$ let $<_\alpha$ denote the restriction of $<$ to $\{\beta : \beta < \alpha\}$.

<u>2.12 Lemma</u>. For each $\alpha \in \Omega^s$ we have

1) $x < y \longrightarrow \alpha[x] \subset \alpha[y]$
2) $\beta < \alpha \longrightarrow \beta \in \bigcup_{x \in N} \alpha[x]$.

Hence $<_\alpha$ is a <u>well-ordering</u> with bottom 0 and $\beta+1 \leq \alpha$ whenever $\beta < \alpha$.

<u>Proof</u>. We prove 1) and 2) simultaneously, by induction over the generation of α in Ω^s:
If $\alpha = 0$ there is nothing to do. If $\alpha = \alpha'+1$ then for each x, $\alpha[x] = \alpha'[x] \cup \{\alpha'\}$ and the induction hypothesis holds for α', so we immediately have 1); and for 2) suppose $\beta < \alpha$. Then either $\beta < \alpha'$ in which case $\beta \in \alpha'[x]$ for some x, or $\beta = \alpha'$. Whichever is the case, we have $\beta \in \alpha[x]$ some x.
If $\alpha = \sup\alpha_n$ then for each x, $\alpha[x] = \alpha_x[x]$, and by structuredness, $\alpha_x \in \alpha[y]$ when $x < y$, and therefore $\alpha[x] = \alpha_x[x] \subset \alpha_x[y] \subset \alpha[y]$.
For 2) suppose $\beta < \alpha$. Then $\beta \leq \alpha_k < \alpha_{k+1}$ some k, so by the induction hypothesis applied to α_{k+1} we have $\beta \in \alpha_{k+1}[x]$ for some $x > k$, hence $\beta \in \alpha[x]$.

To show that $<_\alpha$ is a well-order, little remains to be done. We know already that $<$ is well-founded because of the inductive nature of the generation of each $\alpha \in \Omega$. So we only need check that $<_\alpha$ is a linear ordering, i.e. that whenever $\beta, \gamma < \alpha$ we have either $\beta < \gamma$ or $\beta = \gamma$ or $\gamma < \beta$. But if $\beta, \gamma < \alpha$ then by 1) and 2) there is some $x \in N$ such that $\beta, \gamma \in \alpha[x]$. Clearly $\alpha[x]$ is a finite linear sub-ordering of $<_\alpha$:

$$0 = P_x^n(\alpha) < P_x^{n-1}(\alpha) < \ldots\ldots < P_x^2(\alpha) < P_x(\alpha)$$

so if $\beta \neq \gamma$ then one of them is $<$ the other.

Therefore $<_\alpha$ is a well-order whenever $\alpha \in \Omega^s$. Obviously 0 is its bottom element and if $\beta<\alpha$ then $\beta \in$ some $\alpha[x]$ and the only way this can happen - because of the definition of P_x - is if there is some γ "just above" β in $\alpha[x] \cup \{\alpha\}$ such that $\beta+1 \leq \ldots \leq \gamma \leq \alpha$. Hence $\beta<\alpha \to \beta+1 \leq \alpha$ $\square$

Remark The above lemma is important because it shows that Ω^s singles out those tree-ordinals α whose orderings $<_\alpha$ are exact copies of ordinary set-theoretic ordinals $0 < 1 < 2 < \ldots < \beta < \beta+1 < \ldots < \alpha$. In addition however the ordering has a further structural property which is crucial for the definition of number-theoretic functions by recursion. Namely, each limit point $\lambda = \sup \lambda_x \leq \alpha$ comes equipped with a fixed choice of so-called fundamental sequence approximating to it: i.e. $\lambda_0 < \lambda_1 < \lambda_2 < \ldots < \lambda_x < \ldots\ldots < \lambda$.
Example: let $\lambda = \sup \lambda_x$ where $\lambda_x = x+1$ and $\lambda' = \sup \lambda'_x$ where $\lambda'_x = x+2$. λ, λ' are "order-isomorphic" but incomparable.

2.13 Definitions. For each $\alpha \in \Omega^S$, $x \leq y \in N$, $s : N \to N$,

1) $G_\alpha(x)$ = size of $\alpha[x]$ = least n . $P_x^n(\alpha) = 0$

2) i_{xy} = identity embedding of the numbers less than x into the numbers less than y.

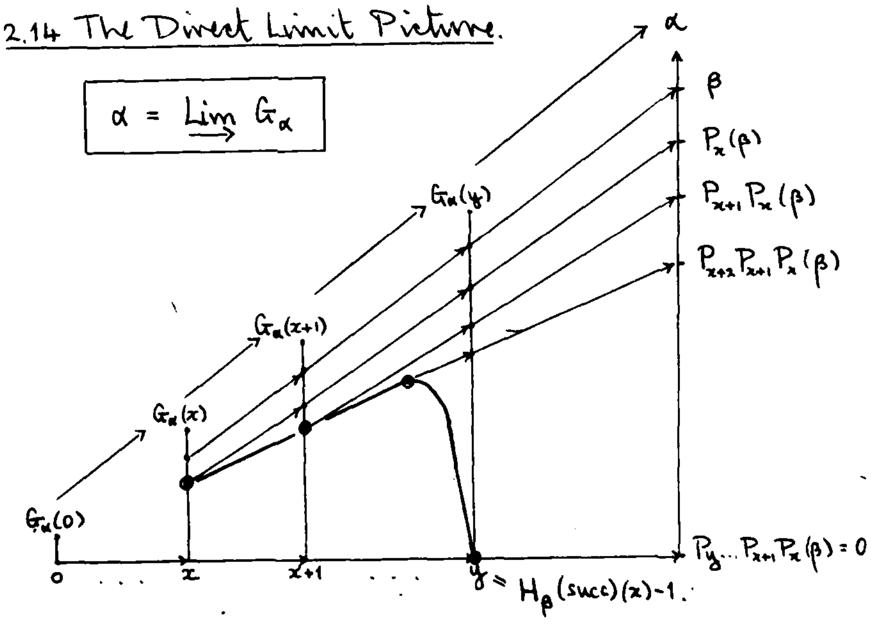
3) $G_\alpha(i_{xy})$ = the finite map from numbers less than $G_\alpha(x)$ into numbers less than $G_\alpha(y)$, given by the rule:
$$G_\alpha(i_{xy})(m) = G_\gamma(y)$$
where $\gamma \in \alpha[x]$ is such that $m = G_\gamma(x)$.

4) $H_0(s)(x) = x$ and for all $\alpha \neq 0$,

$H_\alpha(s)(x) = s^{n+1}(x)$ where n is the least such that $P_{s^n(x)} P_{s^{n-1}(x)} \cdots P_{s^2(x)} P_{s(x)} P_x(\alpha) = 0$.

2.14 The Direct Limit Picture.

$$\boxed{\alpha = \varinjlim G_\alpha}$$


α
β
$P_x(\beta)$
$P_{x+1}P_x(\beta)$
$P_{x+2}P_{x+1}P_x(\beta)$
$G_\alpha(y)$
$G_\alpha(x+1)$
$G_\alpha(x)$
$G_\alpha(0)$
$0 \quad . \; . \; . \quad x \quad x+1 \quad . \; . \; . \quad y$
$P_y \ldots P_{x+1}P_x(\beta) = 0$
$y = H_\beta(succ)(x)-1$.

2.15 Lemma. The function G_α and functional H_α satisfy the following recursive definitions where $\lambda = \sup \lambda_x$:

$$G_0(x) = 0 \quad , \quad G_{\alpha+1}(x) = G_\alpha(x) + 1 \quad , \quad G_\lambda(x) = G_{\lambda_x}(x).$$

$$H_0(s)(x) = x \quad , \quad H_{\alpha+1}(s)(x) = H_\alpha(s)(s(x)) \, , \quad H_\lambda(s)(x) = H_{\lambda_x}(s)(x).$$

Proof immediate from the definitions above.

2.16 Arithmetic on Ω^s.

Addition
$$\alpha + 0 = \alpha$$
$$\alpha + (\beta+1) = (\alpha+\beta)+1$$
$$\alpha + \lambda = \sup(\alpha + \lambda_x)$$

Then $G_{\alpha+\beta}(x) = G_\alpha(x) + G_\beta(x)$; $H_{\alpha+\beta}(s) = H_\alpha(s) \circ H_\beta(s).$

Multiplication (provided $\alpha \neq 0$).
$$\alpha . 0 = 0$$
$$\alpha . (\beta+1) = \alpha.\beta + \alpha$$
$$\alpha . \lambda = \sup(\alpha.\lambda_x)$$

Then $G_{\alpha.\beta}(x) = G_\alpha(x).G_\beta(x)$; $H_{\alpha.\beta}(s) = H_\beta(H_\alpha(s))$

Exponentiation (provided $\alpha \neq 0,1$).
$$\alpha^0 = 1$$
$$\alpha^{\beta+1} = \alpha^\beta . \alpha$$
$$\alpha^\lambda = \sup(\alpha^{\lambda_x})$$

Then $G_{\alpha^\beta}(x) = G_\alpha(x)^{G_\beta(x)}$; $H_{\alpha^\beta}(s) = H_\alpha^{(\beta)}(s)$.

The "first" limit $\quad \omega = \sup(x+1)$ where $x+1 = 0+1 \dots +1$

The "first" epsilon $\quad \varepsilon_0 = \sup(\omega, \omega^\omega, \omega^{\omega^\omega}, \dots\dots)$

We end this chapter with some basic observations concerning recursion on WFRS': $A \subset \Omega$.

<u>2.17 Definition</u> If $\alpha \in \Omega$ is such that $A = \{\beta : \beta < \alpha\}$ forms a WFRS (with the standard predecessors P_x) then we often denote REC(A) instead by REC(α)

<u>2.18 Lemma</u> Suppose $\alpha \in \Omega$ is such that every "limit" $\lambda = \sup \lambda_n \leq \alpha$ has the property $\forall n (\lambda_n \neq 0)$. Then for every $\beta < \alpha$

1) $G_\beta \in$ REC(α)
2) $s \in$ REC(α) $\longrightarrow$ $H_\beta(s) \in$ REC(α)

<u>Proof.</u> G_β and $H_\beta(s)$ are definable by the following "α-recursions":

1) $$G_0(x) = 0$$
$$G_\beta(x) = G_{P_x(\beta)}(x) + 1$$

2) $$H_0(s)(x) = x$$
$$H_\beta(s)(x) = H_{P_x(\beta)}(s)(s(x))$$ □

<u>Note</u> that H is given by a tail-recursion, so it could alternatively be defined by the while-loop

2)' <u>while</u> $\beta \neq 0$ <u>do</u> $\beta := P_x(\beta)$; $x := s(x)$ <u>od</u> .

Note also the difference between G_β and $H_\beta = H_\beta(\text{succ})$

$$G_\beta(x) = \text{succ} \circ G_{P_x(\beta)}(x) \quad , \quad H_\beta(x) = H_{P_x(\beta)} \circ \text{succ}(x).$$

The difference may at first seem slight, but in fact it is far from it! Much interesting work has been generated from analysis of the difference.

2.19 Lemma. If $A \subset \Omega$ forms a WFRS with $\omega \in A$ then $\mathrm{REC}(A)$ is closed under primitive recursion.

Proof. Suppose h is defined from $g_0, g_1 \in \mathrm{REC}(A)$ by a primitive recursion:

$$\begin{cases} h(0,x) = g_0(x) \\ h(z+1,x) = g_1(z,x,h(z,x)) \end{cases}.$$

Define $f : A \times N \times N \longrightarrow N$ by recursion over A:

$$f(0;z,x) = g_0(x)$$
$$f(\alpha;z,x) = \text{if } z=0 \text{ then } g_0(x) \text{ else } g_1(z-1,x,f(P_{z-1}(\alpha);z-1,x)).$$

Putting $\alpha = z$ and noting that $P_{z-1}(z) = z-1$, it is now easy to see by induction on z that
$$f(z;z,x) = h(z,x).$$

Putting $\alpha = \omega$ and noting that $P_{z-1}(\omega) = z-1$, we then have, for all $z, x \in N$,
$$f(\omega;z,x) = h(z,x).$$

Therefore $h \in \mathrm{REC}(A)$ since $\omega \in A$. □

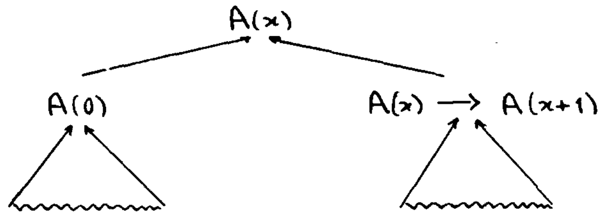
2.20 Theorem. For every true specification $\forall x \exists y R$ there is an $\alpha_R \in \Omega$ such that the specification is satisfied by some function definable in $\mathrm{REC}(\alpha_R+1)$.

Proof Let $\alpha_R = \sup_x y_x$ where $y_x = $ least y. $R(x,y)$. Then $G_{\alpha_R} \in \mathrm{REC}(\alpha_R+1)$ satisfies the specification since for every x, $G_{\alpha_R}(x) = G_{y_x}(x) = y_x$. □

§3. Unravelling Inductive Proofs & Recursions.

To assess the computational complexity of a recursion one needs to "unravel" it somehow, or in other words "simulate" its intended sequence of computation steps. There should therefore be an analogous procedure for measuring the complexity of inductive proofs. There is! – it goes back to Gentzen, was developed by Schütte, then by many others since; and it constitutes one of the principal tools in the proof-theory trade.

Suppose we have a proof of an arithmetical formula $A(x)$, by induction:



We can unravel it into an infinite sequence of "sub-proofs":

$$\begin{tikzcd}
A(0)\ , & , & & A(1) & & , & & A(2) & & , \cdots \\
& & A(0) \arrow[ur, "\text{Cut}"] & & A(0)\to A(1) \arrow[ul] & & A(1) \arrow[ur, "\text{Cut}"] & & A(1)\to A(2) \arrow[ul] & \\
& & & & & A(0) \arrow[ur, "\text{Cut}"] & & A(0)\to A(1) \arrow[ul] & &
\end{tikzcd}$$

where each $A(n)$ requires n nested Cuts.

Therefore we could eliminate the induction entirely, provided we are prepared to allow an infinite proof-rule:

$$\begin{tikzcd}[column sep=small] & & & A(x) & & \\ A(0) \arrow[urrr] & A(1) \arrow[urr] & & A(2) \arrow[u] & \ldots & A(n) \arrow[ull] & \ldots \end{tikzcd}$$

How can we then measure the size and complexity of the resulting infinite proofs?
By using tree-ordinals to represent proof-trees!

The above rule is called the ω-rule and the "semi-formal" system we are about to define is called ω-Arithmetic:

3.1 Definition of ω-Arithmetic (Buchholz).
We are going to define inductively, the notion of an α-bounded infinitary proof $\vdash^{\alpha} \Gamma$ where $\alpha \in \Omega^{S}$.
The finite sets of formulas Γ, and most of the logical rules, will be just like those of PA in §1, except that a new unary relation "$x \in N$" – meaning x is a natural number – is added to the language, together with a new axiom "$0 \in N$" and a new rule saying N is closed under succ.
The new ω-rule is formulated in such a way that it incorporates both Induction and ∀-rules.
This means that we will no longer need to consider formulas containing free variables. All formulas will be "closed" ones.
The tree-ordinals control proof-structure in a strict way using a neat but delicate mechanism due to Buchholz – we call it the Accumulation Rule

Axioms (any $\alpha \neq 0$) $\quad \vdash^{\alpha} \Gamma, A \quad$ where A is any true closed atomic formula of PA.

$\vdash^{\alpha} \Gamma,$ "$0 \in N$"

$\vdash^{\alpha} \Gamma,$ "$m \notin N$", "$m \in N$" $\quad$ for any number m.

Logical rules

($\wedge$) $\quad \vdash^{\alpha+1} \Gamma, (A \wedge B) \quad$ if $\quad \vdash^{\alpha} \Gamma, A$ and $\vdash^{\alpha} \Gamma, B$.

($\vee$) $\quad \vdash^{\alpha+1} \Gamma, (A \vee B) \quad$ if $\quad \vdash^{\alpha} \Gamma, A$ or $\vdash^{\alpha} \Gamma, B$.

($\exists$) $\quad \vdash^{\alpha+1} \Gamma, \exists x A(x) \quad$ if $\quad \vdash^{\alpha} \Gamma, A(n)$ some numeral n.

(Cut) $\quad \vdash^{\alpha+1} \Gamma \quad$ if $\quad \vdash^{\alpha} \Gamma, \neg C$ and $\vdash^{\alpha} \Gamma, C$.

ω-rule $\quad \vdash^{\alpha+1} \Gamma, \forall x A(x)$ if $\vdash^{\alpha} \Gamma, A(n)$ for every n.

N-rule $\quad \vdash^{\alpha+1} \Gamma,$ "$n+1 \in N$" if $\vdash^{\alpha} \Gamma,$ "$n \in N$".

Accumulation rule $\quad \vdash^{\alpha} \Gamma \quad$ if $\quad \vdash^{\beta} \Gamma \quad$ where $\beta \in \alpha[k]$

and $k = \max \{2\} \cup \{3m :$ "$m \notin N$" is in $\Gamma\}$.

Convention: the occurrence of a formula "$m \notin N$" in a set Γ is a declaration of the integer parameter m. In the following we shall often write

$$m_1 \in N, m_2 \in N, \ldots, m_r \in N \ \vdash^{\alpha} \Gamma_0$$

instead of

$$\vdash^{\alpha} m_1 \notin N, m_2 \notin N, \ldots, m_r \notin N; \Gamma_0 .$$

3.2 Definition For any formula A of arithmetic, let A^N denote the formula constructed in the same way, but with all quantifiers relativized to N, i.e. $\forall x B(x)$ is replaced by $\forall x (x \notin N \vee B^N(x))$ and $\exists x B(x)$ is replaced by $\exists x (x \in N \wedge B^N(x))$.

3.3 Completeness Theorem. For every true statement A of arithmetic there is an $\alpha \in \Omega^S$ such that
$$\vdash^{\alpha} A^N .$$

Proof by induction on the logical structure of A.

A atomic: then $A \equiv A^N$ is an axiom.

$A \equiv B_0 \vee B_1$: then one of B_0 or B_1 is true, so by the induction hypothesis we have $\beta \in \Omega^S$ such that either $\vdash^{\beta} B_0^N$ or $\vdash^{\beta} B_1^N$. Therefore by the $\vee$-rule,
$$\vdash^{\beta+1} A^N .$$

$A \equiv B_0 \wedge B_1$: then both B_0 and B_1 are true, so by the induction hypothesis we have $\beta_0, \beta_1 \in \Omega^S$ such that
$$\vdash^{\beta_0} B_0^N \quad \text{and} \quad \vdash^{\beta_1} B_1^N .$$
Note that for any $\gamma, \delta \in \Omega^S$, $\vdash^{\delta} \Gamma$ implies $\vdash^{\gamma+\delta} \Gamma$. Define $\alpha = \sup \alpha_x \in \Omega^S$ where $\alpha_0 = \beta_0 + 1$, $\alpha_1 = \alpha_0 + \beta_1 + 1$ and $\alpha_{n+2} = \alpha_{n+1} + 1$. Then $\beta_0 \in \alpha[2]$ and $\alpha_0 + \beta_1 \in \alpha[2]$ so by the accumulation rule and the above note,
$$\vdash^{\alpha} B_0^N \quad \text{and} \quad \vdash^{\alpha} B_1^N$$
Therefore by the $\wedge$-rule, $\vdash^{\alpha+1} A^N$.

$A \equiv \exists x B(x)$: then for some n, $B(n)$ is true so by the induction hypothesis we have a $\beta \in \Omega^S$ such that $\vdash^{\beta} B^N(n)$. By the N-rule, $\vdash^{\beta+n} n \in N$, and by the accumulation rule, $\vdash^{\beta+n} B^N(n)$. Thus by the $\wedge$-rule,

$$\vdash^{\beta+n+1} n \in N \wedge B^N(n)$$

and by the $\exists$-rule, with n as witnessing term,

$$\vdash^{\beta+n+2} \exists x (x \in N \wedge B^N(x))$$

i.e. $\vdash^{\alpha} A^N$ where $\alpha = \beta + n + 2$.

$A \equiv \forall x B(x)$: then for every n, $B(n)$ is true, so by the induction hypothesis we have $\beta_0, \beta_1, \beta_2, \ldots \in \Omega^S$ such that for every n, $\vdash^{\beta_n} B^N(n)$. Hence by the $\vee$-rule, $\vdash^{\beta_n+1} n \notin N \vee B^N(n)$. Now define $\alpha = \sup \alpha_x \in \Omega^S$ where $\alpha_0 = \beta_0 + 1$ and $\alpha_{x+1} = \alpha_x + \beta_{x+1} + 1$. Then using the "note" above we have, for every n,

$$\vdash^{\alpha_n} n \notin N \vee B^N(n) \quad \text{where } \alpha_n \in \alpha[3n].$$

Therefore by the accumulation rule, for every n,

$$\vdash^{\alpha} n \notin N \vee B^N(n)$$

and then by the ω-rule,

$$\vdash^{\alpha+1} \forall x (x \notin N \vee B^N(x))$$

as required. $\square$

Remark The correspondence A true $\longmapsto \alpha_A \in \Omega^S$ is constructed according to the evidence we have for believing in the "truth-definition" of A. The stronger the evidence, the more structure we ought to infer about α_A. For example what more do we know about α_A if A is a theorem of PA?

3.4 Embedding Theorem. If Γ, with free variables $x_1 \ldots x_r$, is a theorem of PA then for some integer l and all $n_1, n_2, \ldots, n_r \in N$,

$$n_1 \in N, \ldots, n_r \in N \vdash^{\omega \cdot l} \Gamma^N(n_1 \ldots n_r).$$

(See Buchholz-Wainer) $\square$

<u>3.5 Definition</u> A Σ_1^N- formula is one of the form
$$\exists x_1 \exists x_2 \ldots \exists x_r (x_1 \in N \wedge \ldots \wedge x_r \in N \wedge D)$$
where D is built up from variables $x_1, \ldots, x_r, \ldots$ using the given elementary relations $R, \bar{R}$ of PA and the propositional connectives $\wedge, \vee$.

<u>3.6 Definition</u> A closed Σ_1^N- formula is said to be <u>true at m</u> where m is some natural number, if it is true when N is interpreted as the finite set $\{n : 3n < m\}$. A set Γ of closed Σ_1^N-formulas is said to be <u>true at m</u> if any one of its members is true at m.

<u>Note</u> $m < m'$ and Γ true at $m \longrightarrow \Gamma$ true at m'.

<u>3.7 Bounding Theorem</u> (Buchholz)
Suppose that Γ is a set of closed Σ_1^N-formulas and
$$n_1 \in N, \ldots, n_r \in N \vdash^{\alpha} \Gamma$$
where all of the cut-formulas C involved in the proof are Σ_1^N!
Then putting $k = \max(2, 3n_1, \ldots, 3n_r)$ we have
$$\Gamma \text{ is true at } H_{2^\alpha}(succ)(k).$$

<u>Proof</u> For shorthand we will denote the function $H_{2^\alpha}(succ)$ by h_α so that by 2.15, 2.16 we have
(1) $h_0(x) = x+1$, $h_{\alpha+1}(x) = h_\alpha \circ h_\alpha(x)$, $h_\lambda(x) = h_{\lambda_x}(x)$.

Various properties relating to rate-of-growth of these functions will be needed in the course of the proof. These are numbered (2) – (3) and will be verified separately after the proof is complete.

Proceed by induction over the inductive generation of the proof

$$n_1 \in N, \ldots, n_r \in N \vdash^{\alpha} \Gamma$$

Axiom case: if Γ contains an axiom then Γ is automatically true in $h_\alpha(k)$ whatever α is.

$\wedge$-case: if $\Gamma = \Gamma_0, (D_0 \wedge D_1)$ with $\alpha = \beta + 1$, where $n_1 \ldots n_r \in N \vdash^{\beta} \Gamma_0, D_0$ and $n_1 \ldots n_r \in N \vdash^{\beta} \Gamma_0, D_1$ then by the induction hypothesis both Γ_0, D_0 and Γ_0, D_1 are true at $h_\beta(k)$. Hence Γ itself is true at $h_\beta(k)$ and therefore also at $h_\alpha(k)$ because $h_\beta(k) < h_\beta \circ h_\beta(k) = h_\alpha(k)$ by property (2) h_β is strictly increasing.

$\vee$-case: similar to the $\wedge$-case.

$\exists$-case: if $\Gamma = \Gamma_0, \exists x(x \in N \wedge B(x))$ and $\alpha = \beta + 1$ where for some numeral m,

$$n_1 \ldots n_r \in N \vdash^{\beta} \Gamma_0, m \in N \wedge B(m)$$

then by the induction hypothesis, $\Gamma_0, m \in N \wedge B(m)$ is true at $h_\beta(k)$ and hence also at $h_\alpha(k)$ as above. This automatically implies that $\Gamma_0, \exists x(x \in N \wedge B(x)) = \Gamma$ is true at $h_\alpha(k)$.

Cut-case: suppose $\alpha = \beta + 1$ and there is a Σ_1^N cut-formula $C \equiv \exists z_1 \ldots \exists z_s (z_1 \in N \wedge \ldots \wedge z_s \in N \wedge D)$ such that

$$n_1 \ldots n_r \in N \vdash^{\beta} \Gamma, \neg C \text{ \underline{and} } n_1 \ldots n_r \in N \vdash^{\beta} \Gamma, C$$

Now $\neg C \equiv \forall z_1 \ldots \forall z_s (z_1 \notin N \vee \ldots \vee z_s \notin N \vee \neg D)$ and $\vdash^{\beta} \Gamma, \forall z B(z)$ implies $\vdash^{\beta} \Gamma, B(m)$ for every m.

Therefore for all $m_1, \ldots, m_s \in N$ we have

$$n_1 \dots n_r \in N \vdash^{\beta} \Gamma,\ m_1 \notin N \vee \dots \vee m_s \notin N \vee \neg D$$

But $\vdash^{\beta} B_1 \vee B_2$ implies $\vdash^{\beta} B_1, B_2$ so therefore

$$n_1 \dots n_r \in N,\ m_1 \dots m_s \in N \vdash^{\beta} \Gamma, \neg D.$$

Hence by the induction hypothesis,

(*) $\Gamma, \neg D$ is true at $h_\beta(\max(k, 3m_1, \dots, 3m_s))$.

Applying the induction hypothesis also to the second premise of the cut we obtain

$$\Gamma, C \text{ is true at } h_\beta(k)$$

i.e. there are numbers $m_1 \dots m_s$ such that the max of $3m_1, \dots, 3m_s$ is $< h_\beta(k)$ and

(**) Γ, D is true at $h_\beta(k)$.

Therefore for these particular values of $m_1 \dots m_s$ we have, since $h_\beta(k) < h_\beta \circ h_\beta(k) = h_\alpha(k)$,

$$\Gamma, D \text{ is true at } h_\alpha(k)$$

and from (*) since $\max(k, 3m_1, \dots, 3m_s) < h_\beta(k)$,

$$\Gamma, \neg D \text{ is true at } h_\beta \circ h_\beta(k) = h_\alpha(k).$$

The only way that both Γ, D and $\Gamma, \neg D$ can be true at $h_\alpha(k)$ is when Γ itself is true at $h_\alpha(k)$, as required.

ω-case: the ω-rule cannot be applied because it introduces the $\forall$-quantifier, and there aren't any in a Σ_1^N-formula.

N-case: if $\Gamma = \Gamma_0$, "$m+1 \in N$" and $\alpha = \beta + 1$ where

$$n_1 \dots n_r \in N \vdash^{\beta} \Gamma_0, \text{"}m \in N\text{"}$$

then by the induction hypothesis, Γ_0, "$m \in N$" is true at $h_\beta(k)$. So either Γ_0 is true at $h_\beta(k)$, in which case Γ is true at $h_\beta(k)$ and hence at $h_\alpha(k)$; or else "$m \in N$" is true at $h_\beta(k)$ in which case we have $3m < h_\beta(k)$, hence $3(m+1) < h_\alpha(k)$ and hence Γ is true at $h_\alpha(k)$ as required. The proof that $3m < h_\beta(k)$

implies $3m+3 < h_{\beta+1}(k)$ in this case runs as follows. Remember that $\beta \neq 0$. First suppose $\beta = 1$, so that $3m < h_\beta(k) = k+2$ and $(n_1 \notin N, \ldots, n_r \notin N, \Gamma_0, m \in N)$ is an axiom. If $k = 2$ then, since we are dealing with the case where Γ_0 is not true at $h_\beta(k)$, the only possibility is that $m = 0$. If $k \neq 2$ then k must be a multiple of 3. Either way, since $3m < k+2$ we must have $3m \leq k$, and hence $3m+3 < k+4 = h_2(k)$. Now suppose $\beta \neq 1$. Then an easy induction on β gives $h_\beta(k) \geq h_2(k) = k+4$ and therefore
$3m < h_\beta(k)$ implies $3m+3 < h_\beta \circ h_\beta(k) = h_{\beta+1}(k)$.

Accumulation case: suppose $n_1 \ldots n_r \in N \vdash^\beta \Gamma$ where $\beta \in \alpha[k]$. Then by the induction hypothesis, Γ is true at $h_\beta(k)$. Therefore Γ is true at $h_\alpha(k)$ because of property (3) $\beta \in \alpha[k] \longrightarrow h_\beta(k) < h_\alpha(k)$ $\square$

The majorization properties (2), (3) required in the above, follow from the following

3.8 Lemma. Let $s : N \to N$ be strictly increasing, >0. Then for every $\alpha \in \Omega^s$ we have

(2) $H_\alpha(s)$ is strictly increasing

(3) $\beta \in \alpha[x] \longrightarrow H_\beta(s)(x) < H_\alpha(s)(x)$.

Proof. We check (2) and (3) simultaneously, by an induction on α. If $\alpha = 0$ there is nothing to do, since $H_0(s) =$ identity and $0[x] = \phi$. Next suppose $\alpha = \gamma+1$. Then $H_\alpha(s)(z) = H_\gamma(s)(s(z))$ for all $z \in N$ so (2) $H_\alpha(s)$ is strictly increasing since both $H_\gamma(s)$ and s are, and (3) if $\beta \in \alpha[x] = \gamma[x] \cup \{\gamma\}$ then $H_\beta(s)(x) \leq H_\gamma(s)(x) < H_\gamma(s)(s(x)) = H_\alpha(s)(x)$.

Finally suppose $\alpha = \sup \alpha_z \in \Omega^S$. To prove (2): suppose $y < z$. Then $H_\alpha(s)(y) = H_{\alpha_y}(s)(y)$ and by the induction hypothesis applied to α_y we have $H_{\alpha_y}(s)(y) < H_{\alpha_y}(s)(z)$. But $\alpha_y \in \alpha[z] = \alpha_z[z]$ so by the induction hypothesis applied to α_z we have $H_{\alpha_y}(s)(z) < H_{\alpha_z}(s)(z) = H_\alpha(s)(z)$. Therefore $H_\alpha(s)(y) < H_\alpha(s)(z)$ and $H_\alpha(s)$ is strictly increasing. For part (3) notice that if $\beta \in \alpha[x] = \alpha_x[x]$ we can apply the induction hypothesis immediately to α_x in order to give $H_\beta(s)(x) < H_{\alpha_x}(s)(x) = H_\alpha(s)(x)$ □

<u>Note</u> The Bounding Theorem uses the functions

$$h_\beta = H_{2^\beta}(succ)$$

so in order to apply property (3) to the h_β's we need the additional sub-lemma:

$$\beta \in \alpha[k] \longrightarrow 2^\beta \in 2^\alpha[k]$$

This is easily checked by induction on α.

<u>3.9 Corollary to Bounding Theorem.</u>
If a specification $\forall x \in N.\, \exists y \in N.\, R(x,y)$ is provable in ω-Arithmetic by an α-bounded proof in which all cut-formulas are Σ_1^N, then it is satisfiable by a function bounded by:

$$H_{2^\alpha}(succ) \circ K \quad \text{where } K(n) = \max(2, 3n).$$

<u>Proof</u>. From $\vdash^\alpha \forall x (x \notin N \vee \exists y (y \in N \wedge R(x,y)))$ we obtain, by removing "$\forall x$" and "$\vee$",

$$n \in N \vdash^\alpha \exists y (y \in N \wedge R(n,y)) \quad \text{for every } n.$$

So by the Bounding Theorem, for every n there is a $y_n < H_{2^\alpha}(succ)(K(n))$ such that $R(n, y_n)$ □

<u>Question</u> What do we do if the specification proof uses more complex cut-formulas than Σ_1^N ?

<u>Answer</u> Use a Cut-Elimination Theorem to transform the proof into one whose cuts are all Σ_1^N. There is a price to pay however, namely an (iterated) exponential increase in the ordinal bound.

<u>3.10 Cut-Elimination Theorem</u> (Gentzen, ...)
Define the <u>cut-rank</u> of a proof to be the l.u.b. of the lengths of all cut-formulas involved; where length (atomic) = 0, length $(A \wedge B)$ = length $(A \vee B)$ = max (length (A), length (B)) + 1 and length $(\forall x B)$ = length $(\exists x B)$ = length (B) + 1.
Then we have

$$\vdash^{\alpha} \Gamma \text{ with cut-rank } c+1 \longrightarrow \vdash^{\alpha'} \Gamma \text{ with cut-rank } c$$

where α' is any exponential 2^{α} or 3^{α} or ... or ω^{α}.

<u>Proof</u>. See the lectures of Helmut Schwichtenberg.

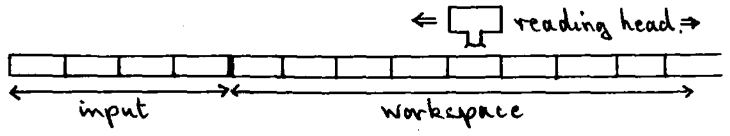
<u>3.11 Corollary</u>. If $\vdash^{\alpha} \forall x \in N.\ \exists y \in N.\ R(x,y)$ with finite cut-rank c, then it is satisfiable by a function bounded by

$$H_{\alpha^*}(\text{succ}) \circ K \quad \text{where } \alpha^* = 2^{2^{\cdot^{\cdot^{\cdot^{\alpha}}}}} \text{ or } \omega^{\omega^{\cdot^{\cdot^{\cdot^{\alpha}}}}} .$$

<u>Proof</u> Apply the Cut-Elimination Theorem to the α-bounded proof, repeatedly until all the cuts are reduced to being Σ_1^N. Then apply 3.9, noting if necessary that $H_{2^\beta} \leq H_{\omega^\beta}$. □

§4. Classifying Provably Recursive Functions.

A recursive function is one which is computable by (say) a Turing machine M which terminates everywhere:



By coding up Turing machine states in a natural way we obtain an "elementary" termination relation:

$T_M(x,y) \Leftrightarrow$ computation of M on input x terminates within y tape-cells.

and an "elementary" value-extractor:

$U_M(x,y) =$ output left on the tape if $T_M(x,y)$.

Thus for every recursive function f there is a T.M. such that the following hold:

1) $\forall x \exists y\, T_M(x,y)$

2) $T_M(x,y) \longrightarrow f(x) = U_M(x,y)$.

In other words, M gives a "computational specification" of f:

$$\forall x \exists y \exists z\, (T_M(x,y) \wedge z = U_M(x,y)).$$

4.1 Definition. $f \in \mathrm{SPACE}(h)$ if there is a Turing machine M which computes f, and (say) a linear function k such that the following holds:

$$\forall x \exists y \leq h \circ k(x).\, T_M(x,y).$$

4.2 Theorem. For every recursive function f there is an $\alpha \in \Omega^s$ such that $\|\alpha\| = \omega$ and

$$f \in \mathrm{SPACE}(H_\alpha(\mathrm{succ})).$$

Proof. Suppose f is computed by a Turing machine which uses no more than $g(x)$ tape-cells for each input x. We can assume that g is strictly increasing. Define $\alpha = \sup \alpha_x \in \Omega^S$ where, for each x, $\alpha_x = 0+1+1+\ldots+1$ ($g(x)$ times). Then every predecessor of α is just an integer, so the set-theoretic ordinal "height" of α is just ω, i.e. $\|\alpha\| = \omega$. Furthermore
$$H_\alpha(succ)(x) = H_{\alpha_x}(succ)(x) = succ^{g(x)}(x) = x + g(x)$$
so $g \leq H_\alpha(succ)$ and $f \in SPACE(H_\alpha(succ))$. $\square$

Remark. Although from 4.2 we have

$$\text{RECURSIVE} = \bigcup_{\substack{\alpha \in \Omega^S \\ \|\alpha\| = \omega}} \text{SPACE } H_\alpha(succ)$$

this is in no way a useful classification of all recursive functions, because in general there is no useful comparison which can be made between one $\alpha \in \Omega^S$ and another, even when they both have $\|\alpha\| = \omega$! There are infinitely-many (in fact continuum-many) different α's in Ω^S, all with $\|\alpha\| = \omega$, but no two of which are comparable under $<$ or any other "reasonable" order.

What we would like to have is a $\sigma \in \Omega^S$ for which

$$\text{RECURSIVE} = \bigcup_{\alpha < \sigma} \text{SPACE}(H_\alpha(succ))$$

but then there could be no way of effectively constructing such a σ, because if there were, $H_\sigma(succ)$ would be a recursive function which dominates all recursive functions (itself included)!

We therefore adopt a more modest aim: given a formal theory $\mathbf{T}$ of arithmetic, construct a tree-ordinal $\sigma_{\mathbf{T}} \in \Omega^s$ such that

$$\bigcup_{\alpha < \sigma_{\mathbf{T}}} \mathrm{SPACE}(H_\alpha(\mathrm{succ}))$$

is precisely the class of all recursive functions which are "provably specifiable" or "provably terminating" in the theory $\mathbf{T}$.

<u>4.3 Definition</u> Call a function <u>provably recursive</u> in a theory $\mathbf{T}$ if it is computed by a Turing machine M in such a way that

$$\mathbf{T} \vdash \forall x \, \exists y \, T_M(x,y).$$

Let $\mathrm{PROV.REC}(\mathbf{T})$ denote the class of all functions provably recursive in $\mathbf{T}$.

The most obvious theory to look at is PA.

<u>4.4 Theorem</u> (Kreisel, Schwichtenberg, Wainer...)

$$\mathrm{PROV.REC.(PA)} = \mathrm{REC}(\varepsilon_0) = \bigcup_{\alpha<\varepsilon_0} \mathrm{SPACE}(H_\alpha).$$

<u>Proof</u> Henceforth we write H_α for $H_\alpha(\mathrm{succ})$.
Recall that $\varepsilon_0 = \sup(\omega, \omega^\omega, \omega^{\omega^\omega}, \dots)$.

1) $\underline{\mathrm{PROV.REC.(PA)} \subset \bigcup_{\alpha<\varepsilon_0} \mathrm{SPACE}(H_\alpha)}$:

Suppose f is provably recursive in PA, so for some Turing machine M which computes f, $\forall x \exists y T_M(x,y)$ is a theorem of PA. Then by the Embedding Thm 3.4 there is a number l such that in ω-Arithmetic,

$$\vdash^{\omega.l} \forall x \in N. \exists y \in N. T_M(x,y).$$

The proof has bounded cut-rank, so by Corollary 3.11 the termination statement $\forall x \exists y\, T_M(x,y)$ is satisfiable by a function bounded by

$$H_{\alpha^*} \circ K \quad \text{where } \alpha^* = \omega^{\omega^{\cdot^{\cdot^{\omega . l}}}} < \varepsilon_0 .$$

But this means that $f \in \mathrm{SPACE}(H_{\alpha^*})$.

2) $\underline{\bigcup_{\alpha<\varepsilon_0} \mathrm{SPACE}(H_\alpha) \subset \mathrm{REC}(\varepsilon_0)}$:

If $f \in \mathrm{SPACE}(H_\alpha)$ then for some linear function k, f is definable explicitly by

$$f(x) = U_M(x, H_\alpha(k(x))) .$$

But U_M and k both belong to $\mathrm{REC}(\varepsilon_0)$ because they are "elementary" and $H_\alpha \in \mathrm{REC}(\varepsilon_0)$ by Lemma 2.18. Therefore $f \in \mathrm{REC}(\varepsilon_0)$.

3) $\underline{\mathrm{REC}(\varepsilon_0) \subset \mathrm{PROV.REC}(PA)}$:

By the correspondence set up in §1, between primitive recursive definitions and proof-rules in the theory of Σ_1^0-Induction, it follows that if f is "elementary" or if f is defined explicitly from functions already known to be provably recursive in PA, then f also is provably recursive in PA.
Suppose then, that $f = f(\bar{\alpha}; \ldots)$ for some fixed $\alpha < \varepsilon_0$, where $f(\alpha; \ldots)$ is defined from given functions $g_0, g_1, \ldots g_m \in \mathrm{PROV.REC}(PA)$, by a recursion:

$$f(0; \underline{x}) = g_0(\underline{x})$$
$$f(\alpha; \underline{x}) = \mathrm{Term}(g_1 \ldots g_m, f_{P(\alpha)}; \underline{x}) .$$

Note that any $\alpha < \varepsilon_0$ is generated from smaller

tree-ordinals $\alpha_1 \ldots \alpha_n$ by $\alpha = \omega^{\alpha_1} + \omega^{\alpha_2} + \ldots + \omega^{\alpha_n}$.
This means that we can represent each $\alpha < \varepsilon_0$ by a number $\bar{\alpha}$, computed in a way reflecting the generation of α. Furthermore the functions $(\bar{\alpha}, x) \longmapsto \overline{P_x(\alpha)}$ will be computable - primitive recursive in fact.
Therefore we can implement the above recursion on a Turing machine M taking input $(\bar{\alpha}; \underline{x})$.
Then proving termination $\forall \underline{x} \exists y\, T_M(\bar{\alpha}, \underline{x}, y)$ clearly amounts to proving $\forall \underline{x}$ ($f(\alpha; \underline{x})$ is defined). Obviously the way to prove this is by transfinite induction up to α:

$$TI(\bar{\alpha}, A) \equiv A(0) \wedge \forall \bar{\beta} < \bar{\alpha} \left(\forall x\, A(\overline{P_x(\beta)}) \longrightarrow A(\bar{\beta}) \right) \longrightarrow \forall \bar{\beta} < \bar{\alpha}\, A(\bar{\beta}),$$

with $A \equiv \forall \underline{x} \exists y\, T_M(\bar{\alpha}, \underline{x}, y)$.
Thus, if we can show that for each $\alpha < \varepsilon_0$ and any formula A, $TI(\bar{\alpha}, A)$ is a theorem of PA, then we will have $f = f(\alpha; \ldots) \in \text{PROV. REC}(PA)$ as required.

First note that $TI(\bar{\omega}, A)$ is just a reformulation of the ordinary principle of induction, which is already built into PA.
Second, note that if we can prove, inside PA,

$$(*) \qquad TI(\bar{\alpha}, A') \wedge TI(\bar{\gamma}, A) \longrightarrow TI(\overline{\gamma + \omega^{\alpha}}, A)$$

where A' is some appropriate formula depending on A, then we can successively generate, inside PA, the principle of transfinite induction up to any given predecessor of ε_0. Then we are finished.

To show why (✱) holds we work very informally "inside" PA:
We will use $TI(\bar{\alpha}, A')$ for an appropriate A', to prove

$$\forall \bar{\beta} < \overline{\gamma + \omega^\alpha} \,.\, A(\bar{\beta})$$

from the assumptions:

(i) $TI(\bar{\gamma}, A)$

(ii) $A(\bar{0})$

(iii) $\forall \bar{\beta} < \overline{\gamma + \omega^\alpha} \left(\forall x \, A(\overline{P_x(\beta)}) \longrightarrow A(\bar{\beta}) \right)$.

<u>Case $\alpha = 0$</u>: From (i), (ii) and (iii) we have $A(\beta)$ for every $\beta < \gamma$. Hence by (iii) we have $A(\gamma)$ also. Thus $\forall \beta < \gamma + 1 \,.\, A(\beta)$ as required.

<u>Case $\alpha \rightarrow \alpha + 1$</u>: Assume true for α. Note that if $\beta < \gamma + \omega^{\alpha+1}$ then $\beta < \gamma + \omega^\alpha . (n+1)$ for some n. Hence the required result will follow if we can prove that for each $n \in N$,

$$\forall \beta < \gamma + \omega^\alpha . (n+1) \,.\, A(\beta).$$

But this is done by ordinary induction on n. For $n = 0$ it amounts to the induction hypothesis on α. For $n+1$ it amounts to $\forall \beta < \gamma' + \omega^\alpha \,.\, A(\beta)$ where $\gamma' = \gamma + \omega^\alpha . n$. This follows from the induction hypothesis on α together with the sub-induction hypothesis on n to change γ into γ'.

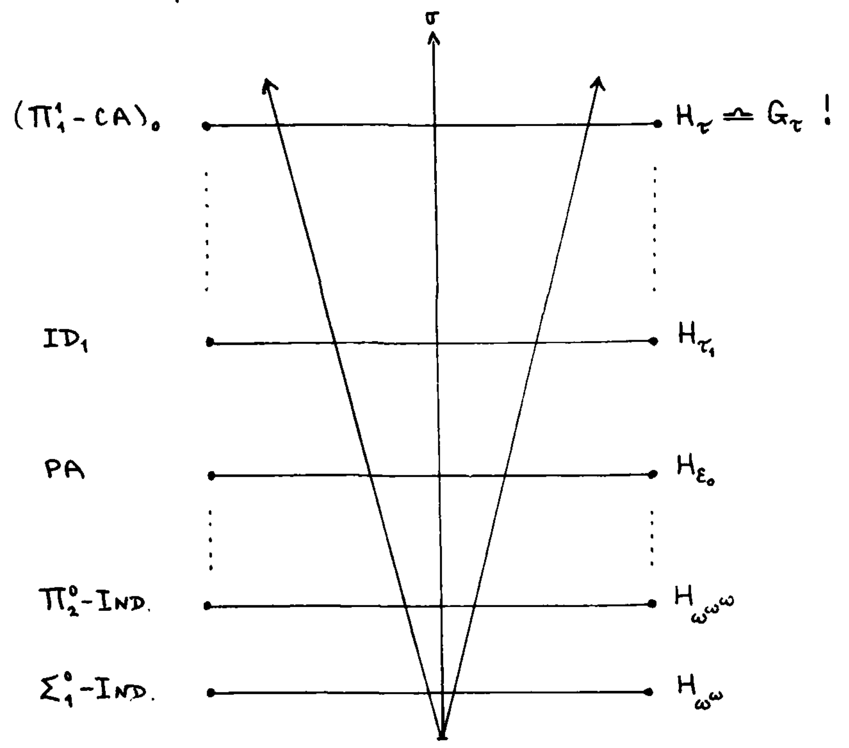
<u>Case $\alpha = \sup \alpha_z$</u>: Assume true for each α_z, $z \in N$. Note that if $\beta < \gamma + \omega^\alpha$ then $\beta < \gamma + \omega^{\alpha_z}$ for some z, so by the induction hypothesis applied to α_z we already have $A(\beta)$. Hence we have $\forall \beta < \gamma + \omega^\alpha \,.\, A(\beta)$ automatically in this case.
This completes the proof of Theorem 4.4. □

Theorem 4.4 is just one in a whole hierarchy of similar results

$$\text{PROV REC}(\mathbf{T}) = \text{REC}(\sigma_{\mathbf{T}}) = \bigcup_{\alpha<\sigma_{\mathbf{T}}} \text{SPACE}(H_\alpha)$$

the crucial element in each case being a careful proof-theoretic analysis of the upper-bound $\sigma_{\mathbf{T}}$ on the "provable ordinals" of the given theory.

4.5 Examples



In each case the termination statement for $H_{\sigma_{\mathbf{T}}}$ is true (by transfinite induction up to $\sigma_{\mathbf{T}}$) but not provable in $\mathbf{T}$ (because $H_{\sigma_{\mathbf{T}}}$ dominates all the provably recursive functions of $\mathbf{T}$).

Remark. For the "small" theories displayed in 4.5, i.e. the subsystems of PA, there is an irregularity in the relationship with the corresponding REC(σ) classes, which turns out to be of some interest. In fact we have

1) $\text{PROV.REC}(\Sigma^0_1\text{-IND}) = \bigcup_{\alpha<\omega^\omega} \text{SPACE}(H_\alpha) = \text{REC}(\omega.\omega)$

2) $\text{PROV.REC}(\Pi^0_2\text{-IND}) = \bigcup_{\alpha<\omega^{\omega^\omega}} \text{SPACE}(H_\alpha) = \text{REC}(\omega.\omega^\omega)$

The correct way to formulate the relationship between REC-definitions and SPACE-complexity is in general as follows (for appropriate β):

$$\text{REC}(\omega.\beta) = \bigcup_{\alpha<\omega^\beta} \text{SPACE}(H_\alpha).$$

This brings us back to the "recursive program"-"while-program" transformation considered in §1.

4.6 Definition. WHILE(α) denotes the class of number-theoretic functions definable by programs built up from the following constructs:

Assignment "$x := \text{given}(x)$"

Conditional "if $R(x)$ then P_0 else P_1"

Composition "$P_0 ; P_1$"

While loop "$\beta := \gamma$; while $\beta \neq 0$ do $\beta := P_x(\beta)$; P od"

for any $\gamma < \alpha$.

<u>4.7 Theorem</u> (Tait 1961, ..., Fairtlough 1989)
Provided that $\{\gamma : \gamma < \beta\}$ satisfies certain basic conditions such as "closure under addition" and "uniform Turing machine representability", we have

$$\mathrm{REC}(\omega.\beta) = \bigcup_{\alpha < \omega^\beta} \mathrm{SPACE}(H_\alpha) = \mathrm{WHILE}(\omega^\beta).$$

<u>Proof</u> (sketch only).
1) First suppose $f \in \mathrm{REC}(\omega.\beta)$, say $f = f(\alpha; \dots)$ where $f(\alpha; \dots)$ is given by recursion as in 2.4 with $\alpha < \omega.\beta$. Implement the recursion on a Turing machine M. Then consider the proof of

$$\forall x \exists y \, T_M(\bar{\alpha}, x, y)$$

by transfinite induction up to α. This proof can be embedded in ω-Arithmetic and the ordinal bound will be (roughly) equal to α. But the cut-rank will be 2 because it's a Π^0_2-formula we're proving. By Cut-Elimination we can then obtain a proof with only Σ^N_1-cuts but the ordinal bound will be something like 2^α. By Corollary 3.9 we therefore have

$$f \in \mathrm{SPACE}(H_{2^\alpha}).$$

But $\alpha \leq \omega.\gamma$ for some $\gamma < \beta$, so therefore

$$H_{2^\alpha} \leq H_{2^{\omega.\gamma}} \leq H_{\omega^\gamma} \cdot h$$

for some "elementary" function h. But then as long as $3 \leq \gamma$ we'll have $h \leq H_{\omega^\gamma}$, so by 2.16

$$H_{2^\alpha} \leq H_{\omega^\gamma} \circ H_{\omega^\gamma} = H_{\omega^\gamma.2} \leq H_{\omega^{\gamma+1}}$$

Hence $f \in \mathrm{SPACE}(H_{\omega^{\gamma+1}})$ and $\omega^{\gamma+1} < \omega^\beta$.

2) If $f \in \mathrm{SPACE}(H_\alpha)$ where $\alpha < \omega^\beta$ then f is

definable explicitly from H_α itself and certain given elementary functions. But $H_\alpha \in \text{WHILE}(\omega^\beta)$ because it is definable by the loop:

<u>while</u> $\alpha \neq 0$ <u>do</u> $\alpha := P_x(\alpha)$; $x := \text{succ}(x)$ <u>od</u>.

Therefore $f \in \text{WHILE}(\omega^\beta)$.

3) Finally suppose $f \in \text{WHILE}(\omega^\beta)$. Say it is definable by a while-loop starting on $\alpha = \omega^\gamma < \omega^\beta$, with sub-program P as in 4.6. Assume that $s_P \in \text{REC}(\omega.\beta)$ is a function which bounds the amount of Turing machine space needed to compute P on input x. Then provided the represent-ation of $P_x(\alpha)$ does not take up "too much" space, we can bound the space needed to compute the while-loop by (guess what?)

$$H_\alpha(s_P)(x) = H_{P_x(\alpha)}(s_P)(s_P(x))$$

Therefore, since $\alpha = \omega^\gamma$,

$$f \in \text{SPACE}(H_{\omega^\gamma}(s_P))$$

and hence f is elementary-definable from $H_{\omega^\gamma}(s_P)$. But again using 2.16, $H_{\omega^\gamma}(s_P) = H_\omega^{(\gamma)}(s_P)$ and H_ω is just the iteration operator:

$$H_\omega(g)(x) = g^{x+1}(x).$$

Therefore $H_\omega^{(\gamma)}(s_P)$ is definable from s_P by $\omega.\gamma$-recursion, i.e. $H_{\omega^\gamma}(s_P) = H_\omega^{(\gamma)}(s_P) \in \text{REC}(\omega.\beta)$ since $\gamma < \beta$. Hence we have shown $f \in \text{REC}(\omega.\beta)$ and this completes the sketch of the proof. □

<u>Remark</u> Thus (in general "pure mathematical" terms) the cost of transformation from recursive- to while-programs is (at worst) an exponential increase in the well-ordering needed to prove termination.

References

[1952] G. Kreisel, On the interpretation of non-finitist proofs II, J. Symb. Logic 17, 43-58.

[1953] A. Grzegorczyk, Some classes of recursive functions, Rozprawy Matem. IV, Warsaw.

[1965] J.W. Robbin, Subrecursive hierarchies, Ph.D. Princeton.

[1968] W.W. Tait, Normal derivability in classical logic, in J. Barwise Ed., Springer Lecture Notes in Math. 72, 204-236.

[1970] S.S. Wainer, A classification of the ordinal recursive functions, Archiv. f. math. Logik 13, 136-153.

[1971] H. Schwichtenberg, Eine klassifikation der ε_0-rekursiven funktionen, Zeit. f. math. Logik 17, 61-74.

[1972] S.S. Wainer, Ordinal recursion, and a refinement of the extended Grzegorezyk hierarchy, J. Symb. Logic 37, 281-292.

[1977] H. Schwichtenberg, Proof theory : some applications of cut-elimination in J. Barwise Ed. Handbook of Mathematical Logic (N. Holland, Amsterdam), 867-896.

[1977] J. Paris and L. Harrington, A mathematical incompleteness in Peano arithmetic, ibid 1133-1142.

[1981] J. Ketonen and R. Solovay, Rapidly growing Ramsey functions, Annals of Math. 113, 267-314.

[1982] L. Kirby and J. Paris, Accessible independence results for Peano arithmetic, Bull. London Math. Soc. 14, 285-293.

[1983] E.A. Cichon, A short proof of two recently discovered independence results using recursion theoretic methods, Proc. Amer. Math. Soc. 87 (4), 704-706.

[1984] H.E. Rose, Subrecursion : functions and hierarchies, Oxford Logic Guides 9, Oxford Univ. Press.

[1984] W. Buchholz, An independence result for $(\Pi^1_1 - CA) + BI$, Annals of Pure and Applied Logic, to appear.

[1984] V.M. Abrusci, J.Y. Girard and J. van de Wiele, Some uses of dilators in combinatorial problems I, II, Research reports in Math., University of Siena. To be published.

[1985] W. Sieg, Fragments of Arithmetic, Annals of Pure and Applied Logic 28, 33-71.

S.S. Wainer,
School of Mathematics,
Leeds University,
LEEDS LS2 9JT.